State-Based Reasoning and Temporal Logic

CS240B Notes

Notes based on Section 10.3 of *Advanced Database Systems*—Morgan Kaufmann, 1997

C. Zaniolo, April 2002

Datalog$_{1s}$

Discrete time, can be modelled using $\mathrm{Datalog}_{1\mathrm{S}}$.
The discrete temporal domain consists of terms built using the constant 0 and the unary function symbol $+1$ (written in postfix notation). For the sake of simplicity, we will write n for

$$(\ldots((0\overbrace{+1)+1)\ldots+1)}^{n\ times}$$

if T is a variable in the temporal domain, then T, $T+1$, and $T+n$ are valid temporal terms, where $T+n$ denotes

$$(\ldots((T\overbrace{+1)+1)\ldots+1)}^{n\ times}$$

Example: The endless succession of seasons

$$
\begin{aligned}
&\mathtt{quarter}(0, \mathtt{winter}). && \\
&\mathtt{quarter}(\mathrm{T}+1, \mathtt{spring}) \leftarrow && \mathtt{quarter}(\mathrm{T}, \mathtt{winter}). \\
&\mathtt{quarter}(\mathrm{T}+1, \mathtt{summer}) \leftarrow && \mathtt{quarter}(\mathrm{T}, \mathtt{spring}). \\
&\mathtt{quarter}(\mathrm{T}+1, \mathtt{fall}) \leftarrow && \mathtt{quarter}(\mathrm{T}, \mathtt{summer}). \\
&\mathtt{quarter}(\mathrm{T}+1, \mathtt{winter}) \leftarrow && \mathtt{quarter}(\mathrm{T}, \mathtt{fall}).
\end{aligned}
$$

Example: Recurring Schedules

Trains for Newcastle leave daily at 800 hours and then every two hours until 2200 hours (military time)

$$
\begin{array}{ll}
\mathtt{before22}(22). & \\
\mathtt{before22}(\mathtt{H}) \leftarrow & \mathtt{before22}(\mathtt{H}+1). \\
\mathtt{leaves}(8, \mathtt{newcastle}). & \\
\mathtt{leaves}(\mathtt{T}+2, \mathtt{newcastle}) \leftarrow & \mathtt{leaves}(\mathtt{T}, \mathtt{newcastle}), \\
 & \mathtt{before22}(\mathtt{T}+2).
\end{array}
$$

Propositional Linear Temporal Logic(PLTL)

PLTL is based on the notion that there is a succession of states $H = (S_0, S_1, \ldots)$, called a *history*.

For instance, Trains to Newcastle can be modelled by a predicate $newcstl$ that holds true in the following states: $S_8, S_{10}, S_{12}, S_{14}, S_{16}, S_{18}, S_{20}, S_{22}$, and it is false everywhere else.

Modal operators are used to define in which states a predicate p holds true.

Propositional Linear Temporal Logic (PLTL).

1. *Atoms:* Let p be an atomic propositional predicate. Then p is said to hold in history H when p holds in H's initial state S_0.

For instance, $\neg newcastl$ is true in our example since it is true in S_0

Temporal Operators

In addition to the usual propositional operators $\vee, \wedge$, and $\neg$, PLTL offers the following operators:

2. *Next:* Next p, denoted $\bigcirc p$, is true in history H, when p holds in history $H_1 = (S_1, S_2, \ldots)$. Therefore, $\bigcirc^n p$, $n \geq 0$, denotes that p is true in history $(S_n, S_{n+1}, \ldots)$.
   For instance,

   $$\bigcirc^8 newcastl \wedge \bigcirc^9 \neg newcastl$$

   is true since there is a train at 8 and no train at 9.

Other Temporal Operators

3. *Eventually:* Eventually q, denoted $\mathcal{F}q$, holds when, for some n, $\bigcirc^n q$.

4. *Until:* p until q, denoted $p\,\mathcal{U}\ q$, holds if, for some n, $\bigcirc^n q$, and for every state $k < n$, $\bigcirc^k p$.

Other Operators

- For instance, the fact that q will never be true can simply be defined as $\neg\mathcal{F}q$.
- The fact that q is always true is simply described as $\neg\mathcal{F}(\neg q))$; the notation $\mathcal{G}q$ is often used to denote that q is always true.
- The operator p before q, denoted $p\mathcal{B}q$ can be defined as $\neg((\neg p)\ \mathcal{U}\ \ q)$—that is, it is not true that p is false until q.

Applications

PLTL finds many applications, including temporal queries and proving properties of dynamic systems.

For instance, the question "Is there a train to Newcastle that is followed by another one hour later?" can be expressed by the following query:

$$?\mathcal{F}\big(\texttt{newcstl} \wedge \bigcirc\texttt{newcstl}\big)$$

Temporal Reasoning with $\mathrm{Datalog}_{1S}$

Every query expressed in PLTL can also be expressed in propositional $\mathrm{Datalog}_{1S}$ (i.e., Datalog with only the temporal argument).

For instance, the previous query can expressed by query $?\texttt{pair\_to\_newcstl}$, where:

$$\texttt{pair\_to\_newcstl} \leftarrow \texttt{newcstl}(\mathtt{J}) \wedge \texttt{newcstl}(\mathtt{J}+1).$$

Temporal Reasoning with $\mathrm{Datalog}_{1S}$

Express $p\,\mathcal{U}\,q$: p must be true at each instant in history, until the first state in which q is true. Use recursion to reason back in time and identify all states in history that precede the first occurrence of `q`.

$$
\begin{array}{ll}
\texttt{post\_q}(\mathrm{J}+1) \leftarrow & \texttt{q}(\mathrm{J}). \\
\texttt{post\_q}(\mathrm{J}+1) \leftarrow & \texttt{post\_q}(\mathrm{J}). \\
\texttt{first\_q}(\mathrm{J}) \leftarrow & \texttt{q}(\mathrm{J}), \neg\texttt{post\_q}(\mathrm{J}). \\
\texttt{pre\_first\_q}(\mathrm{J}) \leftarrow & \texttt{first\_q}(\mathrm{J}+1). \\
\texttt{pre\_first\_q}(\mathrm{J}) \leftarrow & \texttt{pre\_first\_q}(\mathrm{J}+1). \\
\texttt{fail\_p\_Until\_q} \leftarrow & \texttt{pre\_first\_q}(\mathrm{J}), \neg\texttt{p}(\mathrm{J}). \\
\texttt{p\_Until\_q} \leftarrow & \texttt{pre\_q}(0),\ \neg\texttt{fail\_p\_Until\_q}.
\end{array}
$$

Datalog_{1S}–*cont.*

A similar approach can be used to express other operators of temporal logic. For instance, $\mathtt{p}\ \mathcal{B}\ \mathtt{q}$ can be defined using the previous predicates and the rule

$$\mathtt{p\_Before\_q} \leftarrow \mathtt{p}(\mathtt{J}), \mathtt{pre\_first\_q}(\mathtt{J}).$$